Microservices Assignment- 1

Contents

Q1. What is Microservices?

- Microservices is an architectural style where the application code is developed in manageable and small pieces, isolated from others.
- Microservices have their own stack, and function independent of others.
- They communicate with other microservices by REST APIs.
- They are loosely coupled and independently deployable.

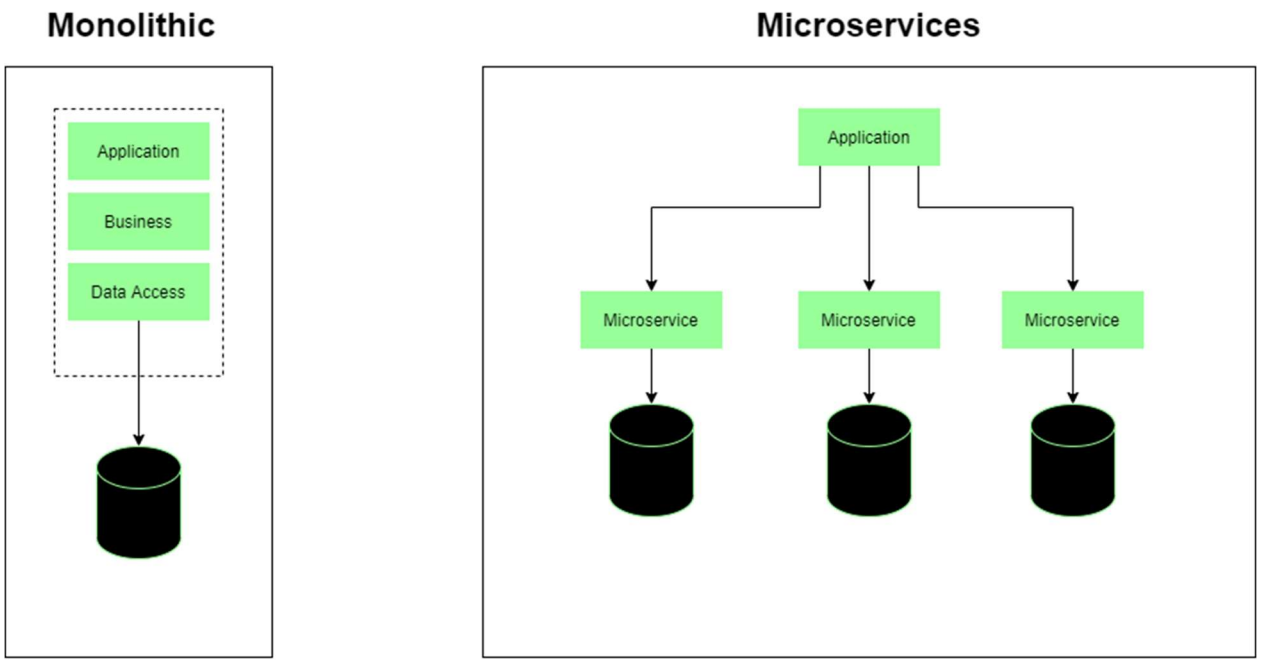

Figure 1.a

Q2. Challenges with Monolithic-oriented architecture

1. Reliability: Bug in one module can potentially break the entire process
    - In monolithic oriented architecture, applications are deployed as a monolith. As a result, multiple identical instances of the application are deployed to the cloud.
    - If one module features a bug, as an example, memory leak, it can break the entire process. Resulting in breaking the functionality of the whole application.
    - As all the instances of the application are the same, the entire application will go down.
2. As the size of the application grows the start-up time can slow down
    - Monolithic application consists of the modular architectural design, but they are still deployed as a single package/monolith.
    - As the application is developed and more features are added, this package that will be deployed grows larger in size.
    - At some point, it becomes large enough to noticeably slow down the start-up time.
    - It also affects the overall build time of the application.
3. Must redeploy the entire application on each code change
    - In Monolithic oriented architecture, as mentioned in the previous point, the application is deployed as a single package.
    - Hence, any code change, be it as small as one line, or as large as an entire module, will result in re-building the application and deploying the entire package again.
    - If the application is large in size, each deployment can cost time.
4. Single Development Stack
    - Monolithic applications are developed using a single development stack.
    - At the time of designing the solution architecture, the technology stack/development stack is decided and the same stack is used throughout the development of the application.
    - This can limit the application from the right tools and technologies which are emerging day by day.
    - Over time, the technology stack needs to be maintained and updated, and if the technology becomes outdated, there's an immediate need of migrating to a latest technology stack, which can be costly.
5. Maintainability
    - A monolithic oriented architecture is used when architecting solutions for one single goal in mind.
    - Sometimes, as the application grows large in size, it's very complex to fully understand and make changes quickly.
    - As a result, there's a chance that the application grows so large that no single developer or group of developers understand the application in its entirety.
    - This can result in developers taking more time to make code changes, each code change resulting in breaking multiple dependencies and extensive manual testing.

Q3. Any three disadvantage and advantage of microservices

Advantages

1. Scalability
   - As microservices are isolated and independent of each other, if demand for certain services grows, we can deploy the particular service across multiple servers to suit the scalability requirements of the application.
   - It is efficient in the sense, that only the services that require to be scaled are scaled, which is cost effective.

2. Maintainability
   - Microservices architecture enable each service to be developed independently by a team of developers to work on the solution.
   - As the microservice is fundamentally independent, the team can focus solely on developing that service.
   - On the other hand, if in future, there's a bug or a feature that needs to be added, only the service that contains the functionality is to be considered.
   - This allows the application to be maintainable over a long period of time.

3. Ability to use different technology stacks
   - Microservices architecture allow each service to exist on its own. The application functions by allowing the microservices to communicate with each other.
   - As a result, we can utilize different technology stacks across services. E.g.: If service A fits better with NodeJS, and Service B suits better with ASP.NET, it's possible to achieve in Microservices architecture.

Disadvantages

1. Complexity
   - To design a Microservices architecture for a solution, adds additional complexity to the project as it is a distributed system.
   - To architect a solution for a Microservices architecture requires experience and the ability to figure out dependencies across services.

2. Deployment
   - Deployment of a microservices-based application is complex.
   - As microservice application consists of a large number of services, each instance needs to be configured, deployed, and monitored.
   - Deployment of each microservice is faster in comparison with monolithic application, but it also adds the complexity of deploying multiple instances of microservices.
   - Manual operations need to be replaced with automation.

3. Testing
   - Testing a microservices application is complex than a monolithic application.
   - In microservices application, testing a particular microservice requires stubs or any other service that it depends upon.

- This can become complex as the dependencies across microservices increase and the communication between the microservices increase.

References:

[1]: Microservices, Wikipedia (https://en.wikipedia.org/wiki/Microservices)

[2]: Monolithic vs Microservices architecture, GeeksForGeeks (https://www.geeksforgeeks.org/monolithic-vs-microservices-architecture/)

[3]: Microservices vs Monolithic Architecture, Mulesoft (https://www.mulesoft.com/resources/api/microservices-vs-monolithic)

[4]: What are Microservices, Smartbear (https://smartbear.com/solutions/microservices/)

[5]: What are Microservices, RedHat (https://www.redhat.com/en/topics/microservices/what-are-microservices)

[6]: Monolithic vs. Microservices Architecture, Microservices (https://articles.microservices.com/monolithic-vs-microservices-architecture-5c4848858f59)

[7]: Monolithic Architecture Pattern, Microservices (https://microservices.io/patterns/monolithic.html)